Discovery protocols for SDN-based Wireless Sensor Networks with unidirectional links

Renan C. A. Alves and Cintia B. Margi

***Abstract*— Ad hoc and wireless sensor networks routing protocols are usually oblivious to the existence of unidirectional links. We propose to use Software Defined Network to take advantage of these links, instead of using the flooding-based or specific-purpose protocols found in the literature. To achieve this goal, it is necessary to devise Controller Discovery and Neighbor Discovery protocols tailored for unidirectional networks. We designed and implemented such algorithms and compared to the traditional Collect-based approach. The results show that SDN is a promising alternative to routing in unidirectional networks.**

***Keywords*— Wireless sensor networks, Software defined networks, Unidirectional networks, Neighbor Discovery**

I. Introduction

Wireless sensor networks (WSN) are a class of wireless communication systems composed of resource constrained devices in terms of processing speed, memory capacity, energy availability, and communication bandwidth [1].

Some of the main applications are environment monitoring and actuation, for example in smart building, precision agriculture and smart cities [1]. In these applications, the WSN devices are usually spread over a large area; therefore, a routing protocol is required to provide end-to-end connectivity. Since these networks are ad-hoc (i.e. not infrastructured), all devices perform routing-related tasks.

Routing protocols design often assume that the physical communication link between two devices is symmetric and bidirectional, i.e. if device A is able to transmit data to device B, device B is able to transmit data to A with same characteristics such as delivery probability. Some examples are AODV [2] and RPL [3].

However, some studies show that this assumption might not be as reasonable as first thought [4], [5]. Many factors lead to link asymmetries such as electromagnetic wave reflection, diffraction and refraction, non-isotropic antennas and variation in manufacturing process.

Routing protocol performance may degrade in the presence of asymmetric links. In particular, performance could be improved if a protocol is able to use unidirectional links. In short, a communication link is said to be unidirectional if it is possible to transmit information in only one direction, i.e. if A can send data to B, but B is not able to directly contact A.

For example, MOLSR-ASYM [6] extends an existing algorithm to detect unidirectional links. It is a link state routing, therefore node reachability information must be disseminated throughout the whole network. Unidirectional Link Counter [7] uses flooding of *route request* and *route reply* messages. It has two versions, the first uses an unspecified ND algorithm, and the other uses flooding to detour unidirectional links, the detour path length should be limited.

Another approach is to provide a bidirectional link abstraction at the MAC layer. BRA [8] hides the fact that there are unidirectional links from the routing layer. The multihop reverse path of unidirectional links is calculated using a reversed Bellman-Ford algorithm. Another work proposes a similar MAC layer abstraction over a network with unidirectional links, by finding reverse paths [9]. Probability-based routing is used to take advantage of temporary links.

Kim et. al [10] studied the case of a main powered basestation node able to transmit packets with high power to the other nodes, and how to perform this transmission reliably. This is a case-specific solution, and does not solve the general unidirectional link problem.

Software Defined Network (SDN) is an alternative routing paradigm, focused on network programmability. Typically, this flexibility is achieved by the centralization of the control plane, thus the routing decisions are made by a centralized controller [11]. Routing rules are informed to the network nodes by a southbound protocol. Specific SDN approaches were proposed for WSN, such as [12], [13], [14].

Our hypothesis is that SDN is able to provide efficient routing in the context of network with unidirectional links. Since the controller has a centralized global view of network links, it can calculate the best route from any source to any destination without flooding the network.

Also, SDN allows flexibly changing routing criteria, since it is altered only on the controller and no assumptions are made about traffic pattern and node radio range.

Nonetheless, SDN needs two underlying protocols, namely a Neighbor Discovery (ND) protocol and a Controller Discovery (CD) protocol. The ND protocol obtains and maintains node neighborhood information, while the CD protocol identifies a next hop candidate to reach the controller. To the best of our knowledge, there is no suitable ND and CD protocols to support SDN on WSNs with unidirectional links.

The main contribution of this work is the design, implementation and evaluation of simple ND/CD protocols tailored for network with unidirectional links. We compare our solution to a traditional ND/CD algorithm that assumes bidirectional links, assessing the metrics of packet delivery, delay, control overhead and time to controller discovery. Finally, we discuss possibilities of enhancements to the simple algorithms, pointing towards future work.

The remaining of this paper is structured as follows:

Renan C. A. Alves and Cintia B. Margi¸ Escola Politécnica – University of São Paulo (USP), São Paulo-SP, Brazil, E-mails: ralves@larc.usp.br, cintia@usp.br.

Section II contains related work, Section III describes our ND and CD algorithms, Section IV contains the experiment methodology, Section V presents the results and Section VI presents final remarks from this paper.

II. Related Work

In this section, we present previous work related to the use of SDN in WSN.

Software defined networks first came up in the domain of cabled networks, represented by the OpenFlow protocol [15]. It assumes that routing devices are different from the end devices, what is not the case in ad hoc networks, in which all end nodes should also route packets. The SDN controller uses the information sent by devices to build a global network view, which is the basis for the centralized route calculation. The SDN controller sends control messages to configure the forwarding table on the devices.

The first attempts to use SDN in WSN tried to adapt OpenFlow [16], [17]. However, these works do not discuss how to adapt the protocol to the underlying medium access and physical layers, whose MTU and data rate are lower than the expected by OpenFlow. Also, ND and CD protocols are not mentioned.

Other noteworthy proposals are TinySDN [12], SDNWise [13] and IT-SDN [14]. These protocols were implemented and tested on networks based on IEEE 802.15.4 MAC layer, and their code is available to download. We now compare them in terms of southbound protocol, ND algorithm and CD algorithm.

The southbound protocol is used to establish communication between the SDN controllers and the network nodes, for example to configure routing rules and to retrieve status information from the nodes. All three protocols implements their own southbound interface, although with similar functionality.

The nodes must inform their neighborhood to the controller, so it can build a global network view. The nodes use an underlying neighbor discovery algorithm to gather this information.

TinySDN uses the Collection Tree Protocol [18] as its ND and CD algorithm. It is already implemented in the target operating system, TinyOS. This algorithm builds a tree rooted at the controller. At first, the other nodes set their distance to the controller as infinity, while the controller broadcasts beacons advertising its existence. Controller neighbors calculate the link status between them, calculating its "rank" using expected transmission count (ETX) as the link quality metric. This information in included in the periodic beacons. The tree is built as each node chooses a parent with minimum rank. The interval between beacons increases if the topology is stable. Unidirectional links are ignored.

SDNWise implements a similar mechanism, but embedded in the southbound protocol, instead of a separate protocol.

Controller discovery is necessary to allow the network to be configured at bootstrap. Both TinySDN and SDNWise perform this task combined with ND, using a tree rooted at the controller. This approach has the downside of not allowing to set a software defined route towards the controller.

IT-SDN has a different approach to ND and CD implementation, since it provides an interface to allow new protocols to be added to the framework. The version available for download also contains a collect-based protocol. However, we chose to use IT-SDN due to the flexibility to change these protocols.

To the best of our knowledge, our work is the first to use SDN to handle unidirectional links in WSN.

III. Simple Discovery Algorithms

This section describes the algorithms proposed in this research and implemented on IT-SDN to support unidirectional links. Dismissing the bidirectional link assumption requires new algorithms, since all that can be assumed from receiving a message is an unidirectional link from the transmitter.

A. A simple Neighbor Discovery Algorithm

The controller is responsible for calculating routes, including the reverse path of unidirectional links, therefore it is possible to design a simple neighbor discovery able to detect all incoming links of each node.

The algorithm consists in transmitting beacons containing the source address at constant intervals. The beacon receiver adds the packet source address to its neighbor table, and trigger an event to send the neighborhood information to the controller. Nodes set the beacon intervals according to their own address to avoid beacon collision.

Figure 1 shows an execution example in a 3-node network with unidirectional links. The rectangles represent the nodes neighbor tables. First, node 1 transmits a beacon and node 2 registers it as neighbor. Next, node 2 transmits a beacon, followed by node 3. The resulting neighbor tables are displayed in the rightmost network snapshot.

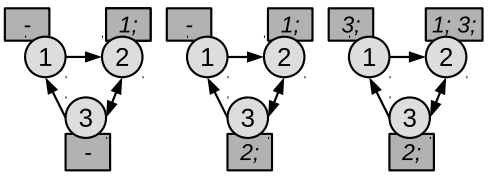


Fig. 1: Simple ND example

The controller is able to build a global view of the network with this information from the nodes. A node learns how to relay the information to the controller using a CD algorithm.

B. A simple Controller Discovery Algorithm

The controller discovery algorithm is used to establish a multihop path from every node in the network to the controller. The route discovery by the CD algorithm may be used throughout all network lifetime or it may be overwritten once the controller is able to communicate with the nodes.

We use an approach similar to distance vector algorithms; however, it was modified to deal with unidirectional link and to find only the controller instead of every route to each other node in the network. To the best of our knowledge, this is the first algorithm with these characteristics.

Every node maintains a table with four columns: *node*, *next hop*, *controller distance* and *number of times the node disseminated this information*. An entry means that a certain

node is able to reach the controller through the *next hop* within *controller distance* hops.

At fixed periodic intervals, nodes broadcast a CD message containing the table[1], if there is at least one entry. If a row has been transmitted a preset number of times, it is removed from the table, thus it is not transmitted anymore.

Upon receiving a CD message, a node updates its table with the new information, for example, if a node is able to reach the controller in less hops. This information is added to the node CD beacons and disseminated in the network. At the beginning, each node sets its distance to the controller as infinity. If the CD message contains the receiving node address, it stores the next hop and the current distance to the controller. Next, it informs the main SDN process about the new controller route, updating the flow table.

The first CD message is sent by the controller after discovering its first incoming neighbors (trough the ND algorithm). Eventually, these nodes learn from CD messages that they are able to reach the controller. Next, these nodes inform their neighborhood information to the controller, which is used to update the controller CD table.

Table I shows a simplified example of a simple 3-node fully unidirectional topology, with the following links: Controller (C) $\Rightarrow$ node 2, node 2 $\Rightarrow$ node 1, node 1 $\Rightarrow$ C. In the first round, the controller discovers that node 1 can reach it. This information propagates to node 2 in round 2 and to node 1 in round 3. In round 4, the controller receives the neighbor information from node 1 and calculates that node 2 can reach it through node 1. Node 2 gets this information in round 5. At this point, all nodes are able to communicate with the controller.

TABLE I: CD algorithm example

Round	Controller		Node 1		Node 2	
	Node	Next hop	Node	Next hop	Node	Next hop
1	1	C				
2	1	C			1	C
3	1	C	1	C	1	C
4	1 2	C 1	1	C	1	C
5	1 2	C 1	1	C	1 2	C 1

IV. Experiments

This section describes our method to execute experiments and the chosen scenarios to pursue the goal of comparing ND and CD algorithms tailored for unidirectional network with algorithms oblivious to this issue.

A. Method

We implemented the algorithms presented in Section III on IT-SDN [14], since its design allows changing the ND and CD algorithms easily. Its source code contains a custom controller software, which we employed in the experiments. The IT-SDN paper contains information about the southbound protocol specification.

The available IT-SDN version relies on hop-by-hop acknowledgements. Since this is not adequate for unidirectional networks, we disabled this feature and implemented end-to-end reliability.

We used COOJA [19], a WSN node emulator with integrated radio medium simulation. COOJA includes several radio medium models, but DGRM (Directed Graph Radio Medium) is most suitable for simulating unidirectional links. This tool is useful to control the network topology, which is hard to achieve in a real testbed.

Each simulation scenario described in the next section is composed of emulated telosB devices (IEEE 802.15.4 compatible) and a controller running on a PC, it communicates with the emulated network through "Serial Server" COOJA plugin. Each scenario was simulated 20 times for 30 minutes, in order to obtain statistical significance.

The following metrics are assessed: (1) delivery, defined as the end-to-end delivery ratio; (2) delay, the time packet takes to reach its final destination (including queue time, and route request-response delay); (3) Control overhead, total number of non-data packets transmitted per node per minute; and (4) the time to controller recognize all network nodes. A control packet retransmission is processed as a new packet transmission for metric calculation purposes.

B. Simulation Scenarios

Three factors were varied in the simulations: topology, network size, and ND/CD protocols. We chose square grid topologies and varied the link status: (1) fully bidirectional, (2) random links disabled[2] or (3) fully bidirectional in which the controller is able to reach all nodes in one hop. The controller was positioned in the corner of the grid, and another node node was set as the data sink.

The chosen network sizes are 9, 16 or 25. The underlying CD and ND protocols used are the algorithms presented in Section III or Collect-based (similar to Collection Tree Protocol briefly explained in Section II). Our proposed ND algorithm do not send link status updates; therefore, to provide a fair comparison, we modified the Collect-based protocol to not send neighbor information messages in case of link state changes, as it would artificially increase the control overhead in comparison to our protocol.

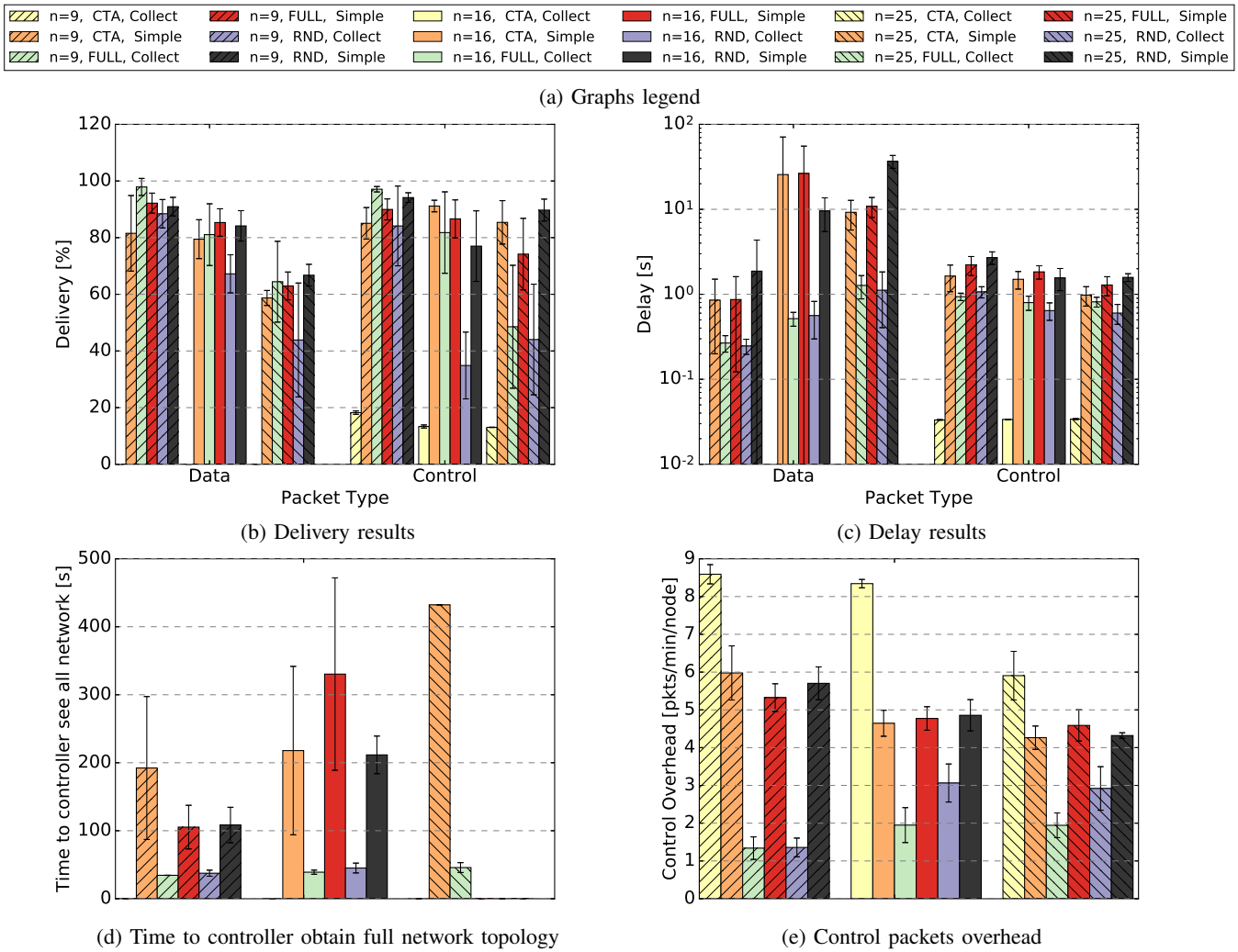
Other parameter had fixed values: regular nodes transmitted data towards the sink at 1 packet/min. In the current implementation our ND algorithm beacon interval is set to 20 + (id mod 10) seconds, and the CD beacon interval is fixed at 10 seconds. Collect protocol parameters are set to Contiki default.

V. Results

Simulations results are presented and discussed in this section. All graphs display 95% confidence intervals obtained

[1]the number of times the node disseminated this information is not included in the beacon message

[2]The same links were disabled for all experiments. The network still had a bidirectional connected component



(a) Graphs legend

(b) Delivery results

(c) Delay results

(d) Time to controller obtain full network topology

(e) Control packets overhead

Fig. 2: Simulation Results

from the simulation runs. The legend for all graphs is displayed in Figure 2a, "n" indicates the network size; CTA (controller to all), FULL (fully bidirectional) or RND (random links off) is the topology type; Collect or Simple is the ND/CD algorithm.

Delivery results are shown in Figure 2b, split into delivery of data packets and of control packets. The most prominent outcome is Collect ND inability to deal with CTA topology, since no data is delivered in any scenario that combine CTA with Collect ND. This algorithm fails to detect some unidirectional links from the controller, causing the controller to set incorrect routes on network nodes.

The fully bidirectional topology shows similar data and control packets delivery. Collect ND scenarios present wider confidence intervals, indicating that this algorithm is more sensitive to small network interferences. In the 25-node scenario, control packet delivery for Collect ND is 60% lower than Simple ND; however, the confidence intervals overlap.

A trend of higher data and control delivery for the Simple ND protocol is observed for the RND topology scenarios. Two scenarios that show a clear advantage of Simple ND regarding control packets delivery are the RND topology for 16 and 25-node scenarios (34% vs. 80% and 44% vs. 89%). We also observed that, for a fixed network size, Simple ND performs similar regardless of the topology, particularly for data packets, while Collect results does not.

Figure 2c exhibits delay results (note that the y axis is log-scale). Collect ND values are better then Simple ND results, both for data and control packets, although some scenarios have overlapping confidence intervals. Simple ND control packets take approximately twice the time to be delivered, while data packets may take from 3.6 (9-node FULL topology) to 50 (16-node FULL topology) times longer.

The reason for such large delay in Simple ND scenarios is that the time to deliver the first data packets is high. It takes longer to disseminate reachability information if no assumption is made about bidirectional link. Since data packets start to be transmitted at 1 minute of simulation time, regardless of network conditions or node knowledge about controller route, the first packet may wait a long time in the node queue until it joins the network and gets a route to deliver the data packet. This behavior is also the cause of the observed wide confidence intervals for Simple ND.

It is interesting to note that CTA topologies allowed the Simple ND algorithm to diminish packet delay in most scenarios. For example, CTA data delivery is 10% faster than FULL topology and 4 times faster than RND topology considering

a 25-node network. Nonetheless, wide confidence intervals hinder drawing definitive conclusions.

Figure 2d contains the time to controller obtain full network topology, these results supports the explanation about delay values. Simple ND takes from 2 to 8 times longer to inform the whole network topology to the controller. In Simple ND, the beacons are sent in fixed periodic intervals, while Collect ND starts with short intervals allowing fast network convergence, increasing the beacon interval later to avoid excessive control overhead. In addition, Simple ND takes longer to propagate controller reachability information, as it is transmitted in periodic CD beacons, while Collect ND assumes bidirectional links, thus is able to set the controller route directly, without this extra delay.

Most 25-node network scenarios were not able to connect all nodes to the controller, thus is displayed as zero in the graph. This is due to the increase of control packets as the network size increases, causing more collisions then the retransmission mechanism is able to cope with. Retransmission timers and beacon intervals could be fine tunned to enhance the results of the 25-node topology.

Finally, we present control overhead results in Figure 2e. Once again, Collect ND adaptive beacon intervals results in lower overall overhead in comparison to Simple ND, specially on smaller networks. The Collect ND performance is also partly due to the modification we made to diminish the number of neighbor information messages, which in turn could trigger route recalculations and increase the number of route configuration messages from the controller. We defer studying neighbor information update frequency to future work.

Nonetheless, as network size increases, this advantage is less prominent. More control traffic increases collisions, making the adaptive Collect ND timer to reset more often, leading to a larger number of control packets per node.

It is interesting to highlight the effect of network topology, as Collect ND tends to perform better in fully bidirectional networks, while Simple ND performs similar in any topology.

Another noteworthy fact is that Simple ND overhead decreased as network size increases, opposed to Collect ND. This is explained from Simple ND design: as the network increases the same CD beacons may be used to disseminate more information at the same time. Note that although the overhead per minute per node decreases, the total number of control packets increases.

VI. Conclusions

Networks with asymmetric and unidirectional links are a reality in WSN domain. We proposed to tackle this problem with an SDN-based approach, in order to avoid flooding-based strategies. To do so, specialized controller and neighbor discovery mechanisms are needed.

We designed, implemented and tested a simple ND/CD algorithm to fill this gap. Although it is simple, the simulation experiments show improvements on packet delivery at the expenses of increased delay and slightly more control overhead.

We consider to have achieved the goal of showing that it is feasible to use SDN to take advantage of unidirectional links.

We intend to improve our proposed algorithms, for example, by dynamically setting beacon intervals, using other metrics than the hop count, and using CD beacons triggered by the controller instead of periodic timers. In addition, we intend to run experiments on larger topologies to test scalability limits.

Acknowledgements

R. C. A. Alves is supported by CNPq PhD fellowship #155372/2016-5. C. B. Margi is supported by CNPq research fellowship #307304/2015-9.

References

[1] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *Computer Magazine*, vol. 37, no. 8, pp. 41–49, 2004.

[2] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561 (Experimental), Internet Engineering Task Force, Jul. 2003.

[3] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Proposed Standard), Internet Engineering Task Force, Mar. 2012.

[4] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of radio irregularity on wireless sensor networks," ser. MobiSys '04. New York, NY, USA: ACM, 2004, pp. 125–138.

[5] L. Sang, A. Arora, and H. Zhang, "On link asymmetry and one-way estimation in wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 6, no. 2, pp. 12:1–12:25, Mar. 2010.

[6] Y. Bai and L. Chen, "Extended multicast optimized link state routing protocol in manets with asymmetric links," in *GLOBECOM*, Nov 2007, pp. 1312–1317.

[7] R. Karnapke and J. Nolte, "Unidirectional link counter - a routing protocol for wireless sensor networks with many unidirectional links," in *Ad Hoc Networking Workshop (MED-HOC-NET)*, June 2015, pp. 1–7.

[8] V. Ramasubramanian and D. Mosse, "Bra: A bidirectional routing abstraction for asymmetric mobile ad hoc networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 116–129, Feb 2008.

[9] X. Chen, Z. Dai, W. Li, and H. Shi, "Performance guaranteed routing protocols for asymmetric sensor networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 1, pp. 111–120, June 2013.

[10] H.-S. Kim, M.-S. Lee, Y.-J. Choi, J. Ko, and S. Bahk, "Reliable and energy-efficient downward packet delivery in asymmetric transmission power-based networks," *ACM Trans. Sen. Netw.*, vol. 12, no. 4, pp. 34:1–34:25, Sep. 2016.

[11] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? implementation challenges for software-defined networks," *Communications Magazine, IEEE*, vol. 51, no. 7, pp. 36–43, July 2013.

[12] B. T. de Oliveira, C. B. Margi, and L. B. Gabriel, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," in *LATINCOM*, Nov 2014, pp. 1–6.

[13] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *INFOCOM*, April 2015, pp. 513–521.

[14] R. C. A. Alves, D. Oliveira, G. N. Segura, and C. B. Margi, "IT-SDN: Improved architecture for SDWSN," in *XXXV Simpósio Brasileiro de Redes de Computadores*, 2017, available at http://www.larc.usp.br/ cbmargi/it-sdn/.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[16] A. Mahmud and R. Rahmani, "Exploitation of openflow in wireless sensor networks," in *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, vol. 1, Dec 2011, pp. 594–600.

[17] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," *Communications Letters, IEEE*, vol. 16, no. 11, pp. 1896–1899, Nov. 2012.

[18] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," ser. SenSys. New York, NY, USA: ACM, 2009, pp. 1–14.

[19] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, Nov 2006, pp. 641–648.